Gravitational Wave Data Analysis : Glitch classification using ML

Name:	**Meet Manoj Maratha**
Registration No./Roll No.:	19190
Institute/University Name:	IISER Bhopal
Program/Stream:	DSE
Problem Release date:	February 02, 2022
Date of Submission:	April 24, 2022

1 Introduction

One of the major topics of this decade has been Gravitational Waves. They are measured when two massive objects which are orbiting each other collapse. They give us some of the critical knowledge of the Universe that we lack.

The data provided contains information about the type of gravitational glitch a non-Gaussian noise transient is detected. A high occurrence rate for these glitches in data can endanger the data we use for finding gravitational waves. There are 22 types of glitches in the data, for which we are trying to prepare a model that can classify these glitches. Figure 1 displays the count plot of all the classes. Here we can see that some of the glitches have a considerably low count in our data. The data is skewed for specific attributes in the data. We can notice this from the Table 1 and the histogram plot of the data in Figure 2. We can observe skewness from the Table 1 when the maximum value of an attribute increases suddenly from its 75% value (we have highlighted such values). We can notice it in Figure 2 by an exponential drop in the count of values in a bin.

Machine learning prefers data to be not skewed as it can make its prediction worse. So to counter it, we scale our data using multiple scaling techniques to check which one gives us the best results. We used Robust Scaler, Quantile Transform (Uniform Distribution), and Quantile transform (Normal Distribution) as our scaling functions. As all these scalers handle outliers admirably, we choose them as it is unclear if there are any outliers in the data. And if they are present in the data, we can account for them, as we can't check for them using plotting the data.

Table 1: Description of data

	GPStime	centralFreq	peakFreq	snr	bandwidth	duration
count	6000	6000	6000	6000	6000	6000
mean	1.13e+9	1527.44	204.57	187.41	2937.87	1.78
std	3.17e+06	1319.21	375.03	1488.28	2663.56	2.68
min	1.13e+0.9	9.78	10.07	7.50	1.26	0.01
25%	1.129e+0.9	256.50	34.18	10.36	425.75	0.22
50%	1.132e+0.9	1228.92	111.13	15.43	2287.86	0.77
75%	1.135e+0.9	2627.85	183.50	37.03	5195.88	2.15
max	1.137e+0.9	4615.13	**2047.11**	**81178.73**	7946.48	**42.16**

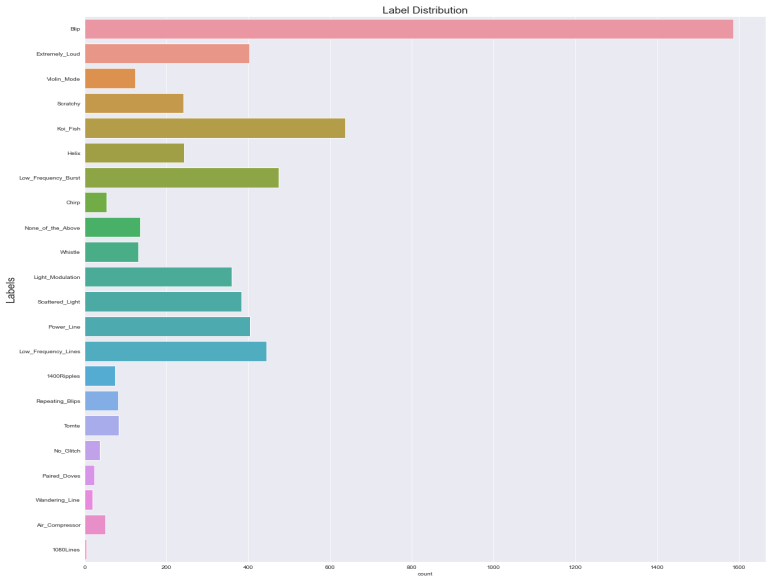

Figure 1: Countplot of different classes

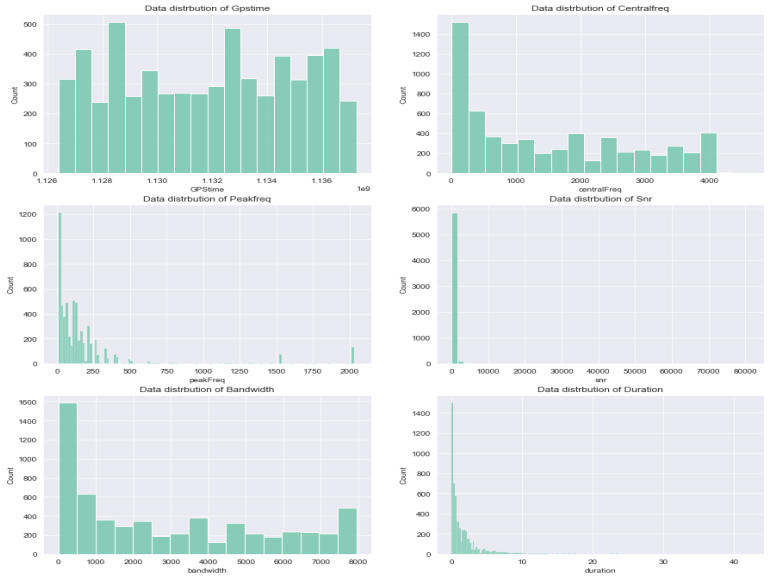

Figure 2: Histogram plot of different attributes

2 Methods

2.1 Data Preprocessing

We need to perform preprocessing on the data before used for building a machine learning model. We have a column named '*ifo*' that contains categorical data. We can encode this information in numerical form by assigning each unique value a number, but this method creates bias between low and high values in the model. One-Hot Encoding is the best way to encode information containing a considerably low number of unique values. In One-Hot Encoding, we create a vector of the size of (no of datapoints*no of rare values of that column). We then set the value of the unique value which appeared for that row as one and everything else as zero. It prevents our machine learning model from being biased to a value. So we One-Hot Encode the *ifo* column.

2.2 Feaute Selection

Feature selection is a process of selection of a subset of attributes. We do this as it does not affect the accuracy of our model much and reduces the cost of computation. Two principal methods implemented in SKlearn that we can use for finding the importance of each attribute are '*mutual_info_classif*' and '*chi2*', which are the functions that calculate Mutual Information [1] of the discrete target variables in the data.

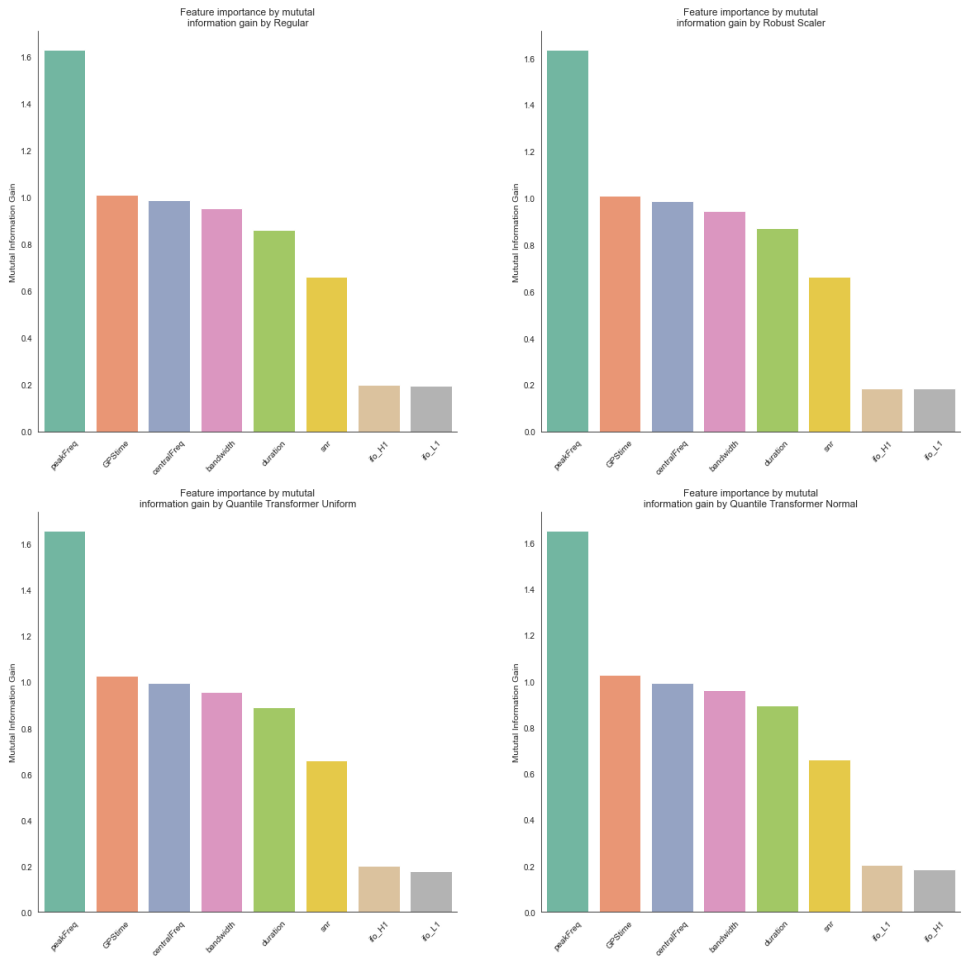

Figure 3: Mutual Info Classification importance for different scalers

Mutual information between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero only if two random variables are independent, and higher values mean high dependency. It depends on a nonparametric method based on entropy estimation from K-Nearest Neighbours distances as described in [1] and [2] which are established on the idea proposed in [3]. The importance of each attribute found using '*mutual_info_classif*' is displayed in Figure 3.

Chi-Squared is a score that calculates chi-squared statistics from the data. The chi-squared test measures dependence between stochastic variables, so using this function highlights features that are highly likely to be independent of class and hence can be dropped. The only handicap of this method is that it works only on positive values. Thus we cannot apply this statistic to Robust Scaled and Quantile Transform (Normal Distribution) as they contain negative values. The importance of each attribute found using '*chi2*' is displayed in Figure 4.
As we notice that the One-Hot Encoded ***ifo*** column is one of the least important attributes in *mutual_info_classif* whereas *chi2* says that it is one of the most important attributes. As they both contradict one other, we decid not to drop it. Hence no columns are dropped from the data.

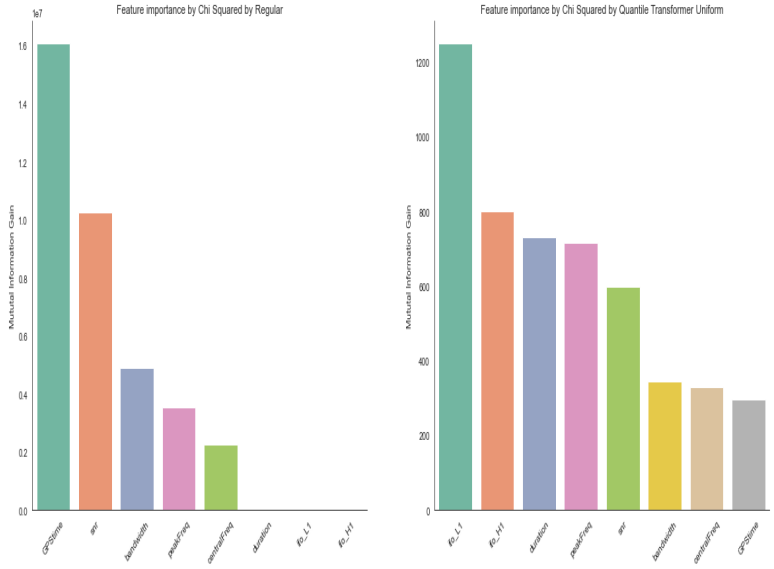

Figure 4: Chi-Squared importance for Quantile Transform (Uniform Distribution)

2.3 Train Test Splitting

First, we drop the *id* attribute from the data as it is just an index for each datapoint. As there are some classes with a considerably low count in the data, we cannot use the simple *train_test_split* function from SKLearn with stratifying option for this data. Thus we use *StratifiedKFold* for splitting the data into five splits, dividing our data in 80-20 fashion, with 80% for training the model.

2.4 Model Training

2.4.1 Random Forest

We perform random forest classification on datasets that have scaled and regular data. In Random Forest Classifier [4] we make '*n_estimatores*' number of decision trees which have a subset of attributes selected from all the features. We then pass data from each of these decision trees and use averaging to improve the predictive accuracy. We train and test the random forest classifier model on all the stratified data created. For hyperparameter tuning, we use *GridSearchCV* function provided in the SKLearn module. Table 2 exhibits tuning parameters and values. The best model in hyperparameter tuning using the metric F1 macro score is selected. F1 Macro is one of the best grading metrics in multi-class classification problems where some classes have low sample sizes.

Table 2: Random Forest Hyperparameter tuning parameters

Parameter Name	Parameter values		
criterion	*gini*	*entropy*	-
max_features	*none*	*sqrt*	*log2*
n_estimators	*10*	*100*	*1000*

2.4.2 Support Vector Clasifier

We perform Support Vector classification on datasets that are scaled. We do not use regular data here as Support Vector machine kernels require data to be centered around zero and have variance in the same order. We train and test the support vector classifier model on all the stratified data created. *GridSearchCV* performs hyperparameter tuning. It is in the SKLearn module. Table 3 exhibits parameters to be tuned and their values. Note that the parameter '*degree*' is used only when the kernel is '*poly*'. Due to diminished combinations, we can supply more values for other parameters

of the support vector classifier. The best model in hyperparameter tuning using the metric F1 macro score is selected. F1 Macro is one of the best grading metrics in multi-class classification problems where some classes have low sample sizes.

Table 3: Support Vector Classifier Hyperparameter tuning parameters

Parameter Name	Parameter values				
kernel	*linear*	*rbf*	*poly*	-	-
degree	*2*	*3*	-	-	-
C	*0.01*	*0.1*	*1*	*10*	*50*

2.4.3 Bagging Clasifier

We perform Bagging classification on datasets that have been scaled and normal. The bagging classifier uses a decision tree classifier as its base. This algorithm has several variations of this method in the literature. Samples drawn with replacement, such as what we are doing, it is known as Bagging [5]. We train and test the bagging classifier model on all the stratified data created. For hyperparameter tuning, we use *GridSearchCV* function provided in the SKLearn module. Table 4 exhibits parameters to be tuned and their values. The best model in hyperparameter tuning using the metric F1 macro score is selected. F1 Macro is one of the best grading metrics in multi-class classification problems where some classes have low sample sizes.

Table 4: Bagging Classifier Hyperparameter tuning parameters

Parameter Name	Parameter values		
max_features	*0.7*	*1.0*	-
n_estimators	*10*	*100*	*1000*
max_samples	*0.7*	*1.0*	-

3 Evaluation Criteria

F1 score is an evaluation metric for classification defined as the harmonic mean of precision and recall. The best value of the F1 score is 1 (perfect accuracy), and the worst value is 0. Precision is the proportion of True positive to the total number of predictions positive for a class. A recall is what proportion of true positives by the sum of true positives and false negatives for a class. F1 score is the combined information of precision and recall of a model for a class. F1 Macro averaged is the unweighted mean of the F1 score of each cateogry. It is a helpful metric in unbalanced class distribution, such as ours, as when a class having low count is classified wrongly by our model, it reflects in the F1 Macro score.

4 Analysis of Results

Table 5 contains analysis results of all the machine learning techniques used with different scalers. In the table, '*Quantile Transform*' means Quantile Transform (Uniform Distribution). We are mentioning the score of models which performed best from all the splits performed by ***StratifiedKFold***. We can see that Bagging Classifier with Quantile Uniform Transform or Quantile Normal transform predicts classes of glitches with high F1 macro score. We can also see that the Bagging classifier is the best machine learning method that we can use for the given data, followed by Random Forest. We can also say that the decision tree type classifier is better suited for the data.

Table 5: Maximum F1 Macro Score of machine learning techniques on Stratified data

Model Name	Scaler			
	Regular	Robust Scaler	Quantile Transform	Quantile Normal Transform
Random Forest	97.0776%	97.0656%	97.0304%	97.0698%
Support Vector Classifier	-	64.4980%	84.2812%	83.0661%
Bagging Classifier	97.2164%	97.1950%	**97.2470%**	**97.2470%**

5 Discussions and Conclusion

A bagging classifier using decision trees is the best model for predicting glitches in gravitational wave data when the data is scaled using Quantile .
In the future, many other Decision tree classifiers can be applied to the data and checked if the F1 Macro score increases. Also, another type of scalers can be applied to the data and used in machine learning classifier models like the Support Vector Classifier, which requires data to be centered around zero and have variance in the same order.

References

[1] Kraskov A. Estimating mutual information. *Physical Review E*, 69, 2004.

[2] Brian C. Ross. Mutual information between discrete and continuous data sets. *PLoS ONE*, 9, 2014.

[3] L. F. Kozachenko. Sample estimate of the entropy of a random vector. *Probl. Peredachi Inf.*, 23(2):9–16, 1987.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.